

DCA0214.1 - LABORATÓRIO DE ESTRUTURAS DE DADOS

Aula 6: Listas sequenciais, encadeadas e suas generalizações

Prof. Felipe Fernandes

12 Abril de 2019

— LEIA COM ATENÇÃO —

- I É proibido utilizar qualquer estrutura de dados ou algoritmos pré-fornecidos por alguma biblioteca C/C++.
 - II Em cada aula desta unidade, as listas de exercícios possuirão questões marcadas com \star . Todas as questões (de todas as aulas) marcadas com \star , devem ser submetidas via SIGAA até às 23h59 do 16 Maio 2019. Peso: 20% na nota de segunda unidade.
 - III O item II não lhe desobriga de resolver as questões sem \star . Todas as questões são importantes para a prova da segunda unidade.
-

- 1. Suponha que temos um conjunto $S = \{s_1, \dots, s_n\}$ com $n < MAX1$ chaves numéricas distintas, onde $MAX1$ é algum limite superior suficientemente grande. Implemente as seguintes estruturas de dados: lista sequencial ordenada, lista simplesmente encadeada ordenada, lista duplamente encadeada ordenada. Você deve implementar os seguintes procedimentos para manipulação dessas estruturas:
 - (a) Busca
 - (b) Inserção
 - (c) Remoção
- 2. \star O Problema da Torre de Hanoi apresenta três pinos ($P1, P2, P3$) e n discos. Inicialmente, todos os discos estão empilhados em $P1$, em ordem decrescente de diâmetro, de baixo para cima. Deve-se mover os discos de $P1$ para $P2$, utilizando $P3$ como auxiliar, de tal modo a respeitar as

seguintes regras: (1) apenas um disco é movido por vez; (2) pode-se mover apenas o disco do topo da pilha de discos; (3) jamais um disco de maior diâmetro é empilhado sobre um menor. Considere o algoritmo recursivo que soluciona a Torre de Hanoi.

Algoritmo 1: Torre de Hanoi

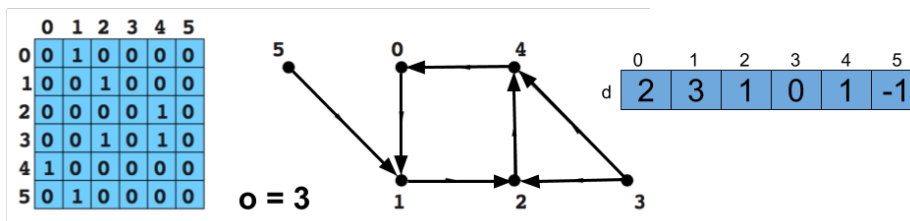
```

1 Procedimento hanoi( $n, P1, P2, P3$ )
2   se  $n == 1$  então
3     | mover o disco do topo do pino  $P1$  para  $P2$ 
4   senão
5     | hanoi( $n - 1, P1, P3, P2$ )
6     | mover o disco do topo do pino  $P1$  para  $P2$ 
7     | hanoi( $n - 1, P3, P2, P1$ )
8   fim

```


Sua tarefa neste exercício consiste em implementar o algoritmo da Torre de Hanoi de modo **iterativo**, utilizando apenas **uma** pilha. Seu algoritmo iterativo deve simular a pilha de recursão que é mantida na versão recursiva. Note que você deve implementar pelo menos três procedimentos: *empilhar* (que insere um elemento na pilha), *desempilhar* (que remove um elemento do topo da pilha) e o *hanoi* (que soluciona a torre de Hanoi de modo iterativo).

3. ★ Suponha que temos n cidades numeradas de 0 a $n - 1$ e interligadas por estradas de mão única. As ligações entre as cidades são representadas por uma matriz A definida da seguinte forma: $A[x][y]$ vale 1 se existe estrada da cidade x para a cidade y e vale 0 em caso contrário. A figura abaixo ilustra um exemplo. Observe que, pela definição acima, **não** há garantias que $A[x][y] = A[y][x]$, para todo x, y . O problema que queremos resolver é o seguinte: determinar a menor distância de uma dada cidade o a cada uma das outras cidades da rede. As distâncias são armazenadas em um vetor d de tal modo que $d[x]$ seja a menor distância de o a x . Se for impossível chegar de o a x , podemos dizer que $d[x]$ vale -1 . Implemente um algoritmo eficiente que, dado a matriz A e uma cidade $0 \leq o < n$, retorne o vetor d de menores distâncias. Dica: use uma fila.



4. ★ Seja um tabuleiro com n -por- n posições, modelado por uma matriz

$A[n][n]$. As posições “livres” são marcadas com 0 e as posições “bloqueadas” são marcadas com -1 . As posições $(0, 0)$ e $(n-1, n-1)$ estão livres. Escreva um algoritmo eficiente que ajude uma formiguinha, que está inicialmente na posição $(0, 0)$, a chegar à posição $(n-1, n-1)$. Em cada posição, a formiguinha só pode se deslocar para uma posição livre que esteja à direita, à esquerda, acima ou abaixo da posição corrente. Seu algoritmo deve imprimir o caminho a ser percorrido pela formiguinha até o destino. Dica: utilize uma fila para achar o caminho de saída e utilize uma pilha para construir (ou recuperar) tal caminho.

	0	1	2	3	4	5
0	 0	-1	0	0	0	-1
1	0	0	-1	0	-1	0
2	0	0	0	0	0	-1
3	0	-1	-1	0	0	0
4	-1	0	0	0	-1	-1
5	0	0	-1	0	0	0

5. A notação polonesa reversa é uma maneira de denotar expressões aritméticas. Nesta notação, os operadores aparecem depois dos operandos correspondentes. Por exemplo, a expressão $((A * B) - (C / D))$ tradicional totalmente parentizada pode ser representada pela expressão polonesa reversa $AB * CD / -$. Utilizando uma estrutura de dados do tipo pilha, implemente:
 - (a) Um algoritmo eficiente que transforme uma expressão da notação totalmente parentizada para notação polonesa reversa (posfixa).
 - (b) Um algoritmo eficiente que transforme uma expressão na notação polonesa reversa numa notação totalmente parentizada.
6. Um deque é uma lista linear que permite a inserção e a remoção de elementos em ambos os seus extremos. Implemente quatro funções para manipular um deque: uma que realiza a inserção de um novo elemento no início do deque, uma que realiza a inserção de um novo elemento no fim do deque, uma que realiza a remoção de um elemento no início do deque e uma que realiza a remoção de um elemento no fim do deque. Utilize uma lista duplamente encadeada.